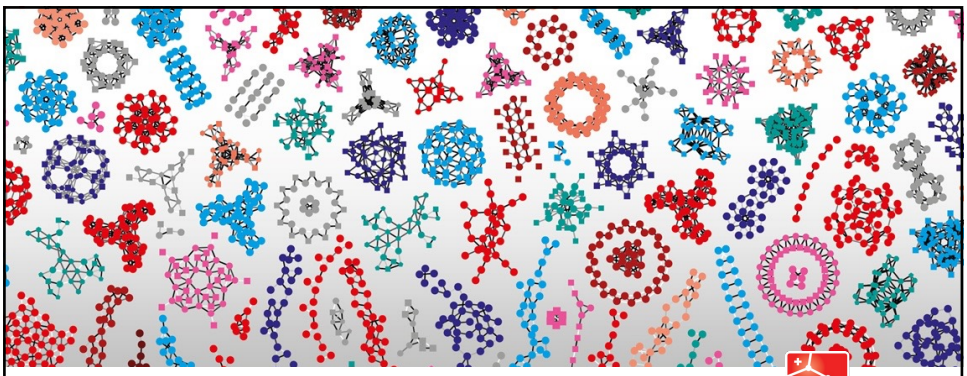


Swiss Institute of Bioinformatics

First Steps with UNIX for Life Scientists

Grégoire Rossier, Robin Engler, Sina Majidian

27 September 2022

1

Learning outcomes

- To get familiar with the UNIX environment
- To use the most common UNIX commands
- To acquire skills necessary for further courses requiring UNIX, like HPC and NGS analysis courses.

2

Outline

1. What is UNIX and why should biologists use it?
2. UNIX filesystem: navigation and usage
3. Environment, processes & Jobs
4. Working with directories and files
5. Working with file content

Exercises are integrated in chapters

An optional exam will be put online at the end of the course

3

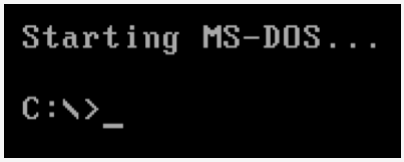
What is UNIX and why should biologists use it?

Swiss Institute of Bioinformatics

4

Command Line Interface (CLI)

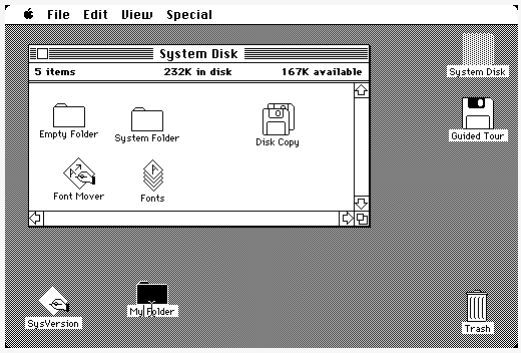
In the 70's, all systems used a command line interface



5

Graphical User Interface (GUI)

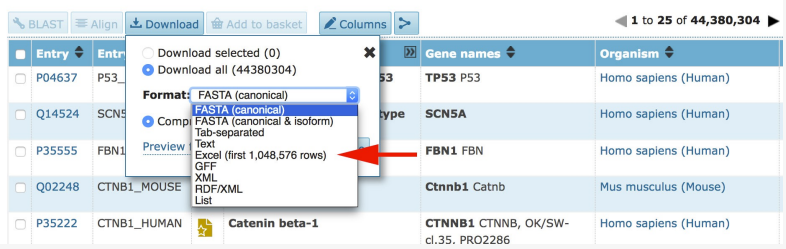
- Development since the 60's
- First popular product was the Macintosh in **1984**
- **Different way of thinking** / acting: windows, menus, icons, mouse and clicks (now touchscreens)



6

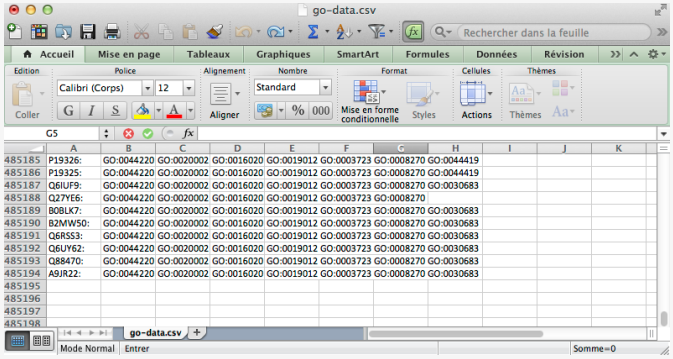
CLI vs GUI – size issues

Dealing with more than 1 million lines



7

CLI vs GUI - complex queries



- How many Swiss-Prot Identifiers are in the file ? ✓
- How many different GO annotations are there ? ✗
- What are the 4 most common GO annotations ? ✗

8

CLI vs GUI - complex queries

- *How many Swiss-Prot identifiers?*

```
userA$ wc -l go-data.csv
485194
```

- *How many different GO annotations?*

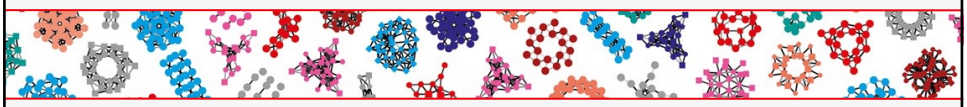
```
userA$ tr " " "\n" < go-data.csv | grep ^GO: | sort | uniq | wc
-l
15696
```

- *What are the 4 most common GO annotations?*

```
userA$ tr " " "\n" < go-data.csv | grep ^GO: | sort | uniq -c |
sort -nr | head -4
127303 GO:0005737
82740  GO:0005524
66591  GO:0016021
52952  GO:0046872
```

9

Outline

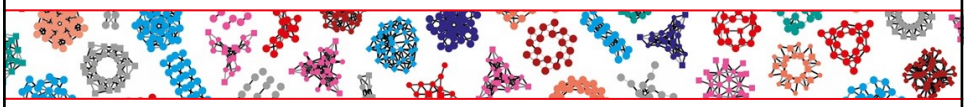


1. What is UNIX and why should biologists use it?
2. UNIX filesystem: navigation and usage
3. Environment, processes & Jobs
4. Working with directories and files
5. Working with file content
6. Optional exam

Exercises are integrated in chapters

10

UNIX filesystem: navigation and usage



SIB Swiss Institute of Bioinformatics

elixir SWITZERLAND

11

Files & directories

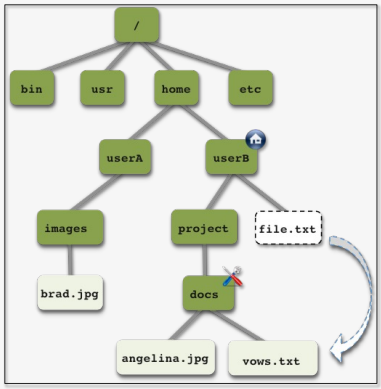
- Organised as an inverted tree
- Single root '/' at the top.

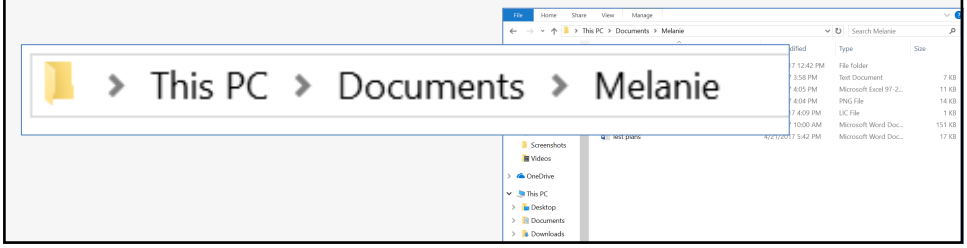
Directory: is a file containing file & directory names

Home directory: is the 'private' directory of a user

Working directory: is the current directory.
The one you are in when executing a command.

File: is a file

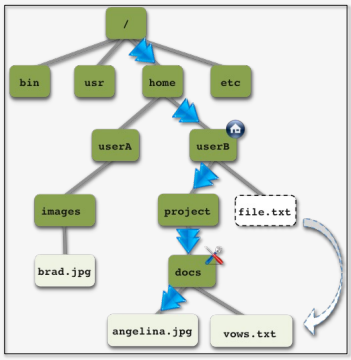




12

Paths

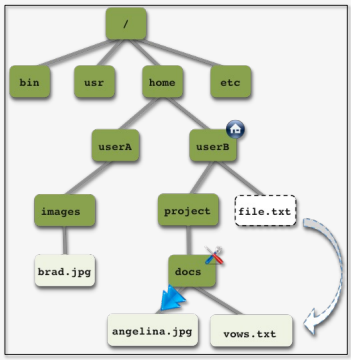
absolute path…



from the root

`/home/userB/project/docs/angelina.jpg`

relative path…



from the working directory

`angelina.jpg`

13

General command syntax

space space

`command -options arguments`

`command`	**program** performing an action
`-option(s)`	modifies the command
`argument(s)`	identifies data (file or directory, with or without a path)

By default, commands are executed in the working directory

14

Help about commands & options

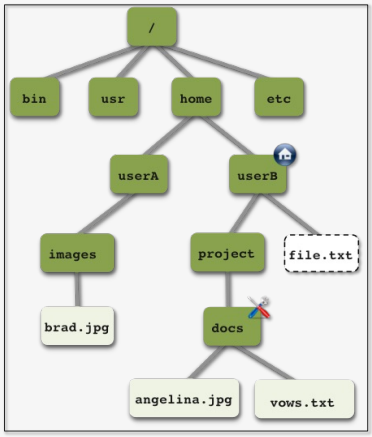
man

```
PWD(1)                    BSD General Commands Manual                    PWD(1)

NAME
     pwd -- return working directory name

SYNOPSIS
     pwd [-L | -P]

DESCRIPTION
     The pwd utility writes the absolute pathname of the current working directory to the stan-
     dard output.

     Some shells may provide a builtin pwd command which is similar or identical to this util-
     ity.  Consult the builtin(1) manual page.

     The options are as follows:

     -L      Display the logical current working directory.

     -P      Display the physical current working directory (all symbolic links resolved).

     If no options are specified, the -L option is assumed.

ENVIRONMENT
     Environment variables used by pwd:

     PWD  Logical current working directory.

EXIT STATUS
     The pwd utility exits 0 on success, and >0 if an error occurs.

SEE ALSO
     builtin(1), cd(1), csh(1), sh(1), getcwd(3)
:
```

Type q to exit

15

Navigation - Locate



pwd

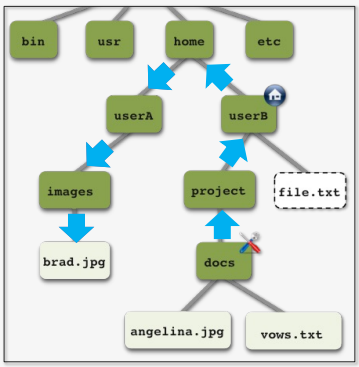
- for 'print working directory'
- indicates the absolute path of the current directory

```
userB$ pwd
/home/userB/project/docs
```

16

Current & parent directories

Example: the current directory is 'docs'



Absolute path to brad.jpg

```
/home/userA/images/brad.jpg
```

Relative path to brad.jpg

```
../../../userA/images/brad.jpg
```

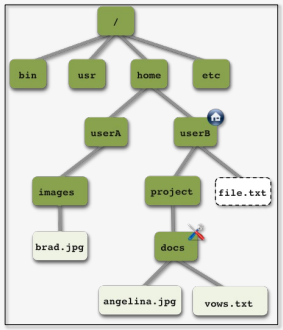
Parent directory

Copy brad.jpg to current directory

```
../../../home/userA/images/brad.jpg .
```

17

Navigation



cd

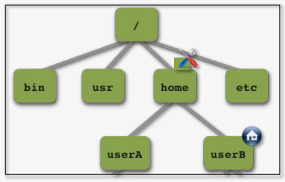
- for 'change directory'
- allows to navigate through the directories

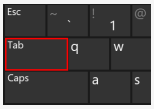
- To go to the root: `cd /`
- To go to user home directory: `cd ~` or just `cd`
- To go up: `cd ..`
- To go down: `cd project/docs`

18

Navigation – Auto-completion

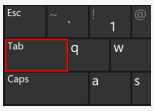
Current working directory: home



[command] + 1x 

-> auto-completion until a ''choice''.

- Example:
  userA
  userB

- [command] + 2x 

-> lists the content of the current directory

19

Navigation – Auto-completion

- Fast navigation
- Safe commands

20

Navigation – Content listing

ls

- for 'list'
- displays the directories and files of the current directory or of a specified directory

Options

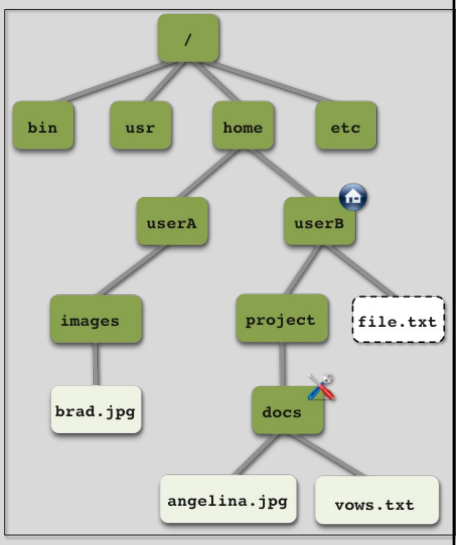
- **-l** shows one item per line, with **detailed information**, such as dates and access rights
- **-h** prints file and directory sizes in **human readable format** (combined with -l)
- **-a** shows all files, including **hidden files**
- **-t** sorts by **modification time** (most recent first), default sorting being by name
- **-r** **reverts** sorting
- **-R** shows content recursively

21

Quiz #1

Current working directory: 'docs'. How would you refer to the file 'brad.jpg' in the 'images' directory?

- ❑ /home/userA/images/brad.jpg
- ❑ ../../../../../../home/userA/images/brad.jpg
- ❑ ../../../userA/images/brad.jpg
- ❑ /home/userB/../userA/images/brad.jpg



22

Warm up - First commands

- **pwd** to print the path of the working directory (*Where am I in the tree?*)
- **ls and cd** to navigate through the directories and list their content (e.g. navigate to *practicals* folder)
  - List directories' content with the options: **-l** , **-a** , **-h** (and -lh) , **-t** (and -lt) , **-r** , **-R** , **~**
  - Explore the directories (cd) with the arguments **..** , **.** , **/** , **~** , **[enter]**

23

Rules – case sensitivity

The command line differenciates **upper** from **lower** case letters:

brad.jpg

≠

BRAD.jpg

≠

brad.**JPG**

24

Rules – filename limitations

Nearly no limitation in length or characters. However…

- forbidden character: **/**
- not recommended characters:
  - characters with a particular meaning for the shell: **space , ; : * ? & % $ | ^ ~ ' " () [] {} ! \ #**
  - international characters: **é à æ ñ ç**
- safe characters: **- _ . 0-9 a-z**

25

Rules – filename extensions

File extensions are **arbitrary** and **do not have a particular meaning** for the operating system, but are useful for the users.

.sh	shell scripts
.pl	perl scripts
.py	python scripts
.txt	text files with no particular format
.csv	text files with **C**omma-**S**eparated **V**alues
.fas	files containing sequences in *FASTA* format*

**FASTA is a text-based format for representing either nucleotide or peptide sequences.*

26

Rules – filename extensions

A file can have several extensions…

archive.**tar**.**gz**

tar = archive
gz = compression

✓

archive.**od**.**cw**

od = no meaning
cw = no meaning

…or no extension

archive

27

Users and permissions

Every file / directory has access properties

who can access? 3 levels of classes:

user	the file owner
group	other users in the same group as the file owner
others	all other users in the system
all	all users in the system

28

Users and permissions

Every file / directory has access properties

what kind of access? 3 access modes:

read	permission to view the content of the file
write	permission to edit the content of the file
execute	permission to run a file (scripts, programs) or to enter a directory

29

Users and permissions

Every file / directory has access properties

what **restriction/permission**?

+	adds the permission
-	removes the permission
=	applies the specified mode to the specified classes (faster if heterogeneous initial modes)

Root or system administrator has all rights!!!

30

Permission modifications

chmod

```
chmod [u g o a] [+ - =] [r w x] file_name
```

user
group
others

r read
w write
x execute

Option:
-R recursive action

chmod can be run by root or by file owner only

31

Permission modifications

```
-rwxrwxrwx file01.txt
-rwxrwxrwx file02.txt
```

How to allow everyone to read these files, but prevent 'group' + 'others' to 'write' and 'execute' them?

```
chmod go-wx file01.txt file02.txt
```

```
-rwxr--r-- file01.txt
-rwxr--r-- file02.txt
```

32

Permission modifications

```
-rwxrwx--- file01.txt
-rwxr-xr-x file02.txt
```

How to allow everyone to read these files, but prevent 'group' + 'others' to 'write' and 'execute' them?

When several files with different permissions, it will be useful to set exact permissions using '='

```
chmod u=rwx,go=r-- file01.txt file02.txt
```

```
-rwxr--r-- file01.txt
-rwxr--r-- file02.txt
```

33

Permission modifications

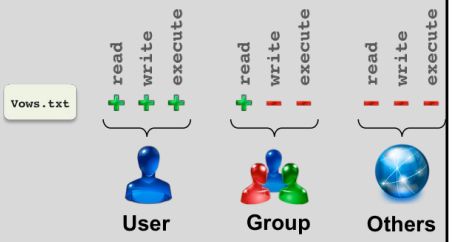
Each configuration of permission can be replaced by a code.
For example ***chmod 700*** = all permissions for owner, but none for all others
ugo

Number	Octal Permission Representation	Ref
0	No permission	---
1	Execute permission	--x
2	Write permission	-w-
3	Execute and write permission: 1 (execute) + 2 (write) = 3	-wx
4	Read permission	r--
5	Read and execute permission: 4 (read) + 1 (execute) = 5	r-x
6	Read and write permission: 4 (read) + 2 (write) = 6	rw-
7	All permissions: 4 (read) + 2 (write) + 1 (execute) = 7	rwx

34

Quiz #2

Given the permissions of the file vows.txt, check all correct statements



- ❑ The root user, who does not belong to the same unix group as the owner, is not allowed to display the file
- ❑ Users belonging to the same unix group as the owner can see/display the file
- ❑ The owner of the file is granted all permissions
- ❑ Users belonging to the same unix group as the owner can modify the file, but a warning is sent to the owner
- ❑ Users belonging to the same unix group as the owner can modify the file, but they have to do a copy first

35

Quiz #3

The following file permissions -rwxr-xr-- mean:

- ❑ read, write and execute by owner
  read and execute by group
  read by others
- ❑ read and write by owner
  execute and read by group and others
- ❑ read, write and execute by owner
  read and execute by group and others as well

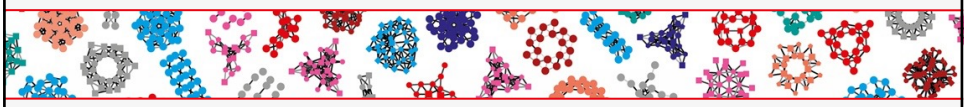
36

In a nutshell

- Files and directories are organised as a tree
  - Navigate along its branches with cd
  - List folders content with ls
- Commands are executed at the current working directory if a path is not specified
- Rules
  - Case sensitivity
  - Filename extensions are arbitrary
  - Characters with a meaning for the shell are not recommended
- Access permissions for reading, writing and execution

37

Outline

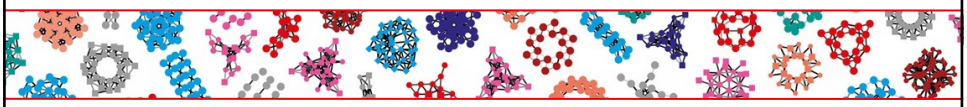


1. What is UNIX and why should biologists use it?
2. UNIX filesystem: navigation and usage
3. Environment, processes & Jobs
4. Working with directories and files
5. Working with file content
6. Optional exam

Exercises are integrated in chapters

38

Environment, processes & Jobs



Swiss Institute of Bioinformatics

39

User environment

man

Equivalent to help, often used to search commands' options

uname

Prints information about the system (several options)

whoami, id, groups, who

Prints information of the user

40

User environment

clear

This command allows to clean the screen

history

Allows to view the history of previous commands, which are listed with an increasing number

Call the last command with: !!

Search a string with ctrl + r

Browse history with up and down arrows

View page by page history with history | less

Browse through the last commands with the UP and DOWN keys

41

Alias

Regularly used complex commands can be renamed for simplification

Example:

alias la="ls -altr"

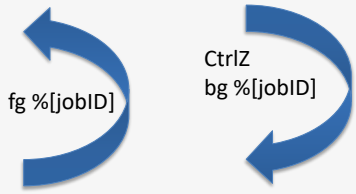
(list the content, including hidden files, ordered by reverse date)

- active only during the current session
- write it in a special file to make it permanent

42

Job status

- Run in the foreground
  - **prompt not available** until the job ends
  - the **output** is **displayed** on the terminal
- Run in the background
  - add **&** after the command
  - the **prompt remains available**
  - the output is **not displayed** on the terminal
- Done
- Stopped
- Terminated / killed -> (CTRL -c, kill)
  - when a process does **not behave as expected** - frozen, gone into a loop...
  - Use **CTRL -c** when in foreground to kill the current process
  - Use **kill %[jobID]** to kill a job in background



43

In a nutshell

- Commands to get information about you and your environment
- History of the last previous commands
- Simplfy commands with alias
- Jobs can be running in foreground or background
- Kill a process / job with CTRL -c when you get stuck

44

Outline

1. What is UNIX and why should biologists use it?
2. UNIX filesystem: navigation and usage
3. Processes & Jobs
4. Working with directories and files
5. Working with file content
6. Optional exam

Exercises are integrated in chapters

45

Working with directories and files

Swiss Institute of Bioinformatics

46

Wildcard characters

are very useful for multiple files manipulation

***** matches **any number of characters** in a file or directory name

```
ls image*.jpeg    ls *.jp*g
```

? matches **exactly one character** - use '??' to match any group of 2 characters

```
ls *.???
```

[ ] specify a **range of characters** allowed at that position – separated by an hyphen; use '!' to exclude a range of characters at that position

```
ls [a-f]* ls [!a-f]*
```

{ } specify a **list of terms** - separated by commas

```
ls *{.html,.txt}
```

47

Exercise 1 – Wildcard characters

Go to directory *practicals/sandbox/* and

- List files names starting with character ‘x’
- List files names containing character ‘x’
- List file names containing an 'x' followed by one random character, followed by an ‘r’
- List file names containing an x and ending with any letter between 'a' and 'c' (example: axona or exonic)

48

Directories & files organisation

mkdir

- "make directory": **creates** a new directory in the current directory
  mkdir new-directory(ies)

Option

-to create:

- several levels at a time: option –p
- several directories at a time with a path:
  mkdir level1/{level2a,level2b,level2c}
- several level with several directories
  mkdir -p l1/{L2a,L2b,L2c}

49

Directories & files organisation

mv

- **moves** a file or a directory to a new location
  mv file-or-directory-name new-location/

- **renames** a file or a directory
  mv file-or-directory-name new-name

Important: Moving files can overwrite existing files. -i option ask before overwritting

50

Directories & files organisation

`mv file1 file2`

If no file2, file1 is renamed
If file2 exists, it's overwritten -> use option -i for confirm.

`mv file1 directory2/`

file1 is moved to directory2

`mv file1 directory2`

file1 is renamed directory2

Good Practice: autocompletion is useful to check if the new name or the new location exist

51

Directories & files organisation

cp

- to **make a copy** of a file with another name
  `cp file-name.txt file-name2.txt`
- to **make a copy of** a file in another directory (same name is kept)
  `cp file-name.txt Other_directory/`
- to **copy a directory** and all its files
  `cp -R directory/ new-directory/`

Important: Copying files can overwrite existing files. Use option -i

52

Directories & files organisation

rm

- to remove/**delete a file**
  `rm file-name.txt`

rmdir

- to remove/**delete a directory**
  `rmdir empty-directory-name/`

Important: by default a directory must be empty before it can be deleted.

53

Directories & files organisation

rm options

- **-R** or **-r** (for 'recursive')
  removes directories and their contents recursively
  `rm -R non-empty-directory-name/`

- **-f** (for '-force')
  never prompts for user confirmation, when files are write-protected

Use these options with care as there is no way back!

Hint: before executing a *rm* command with file names containing wildcard characters, simulate it first with the *ls* command as a control.

54

Quiz #4

What command(s) make(s) a copy of the file myfile.txt in the directory two levels up?

- ❑ cp myfile.txt-2/
- ❑ jump -2 myfile.txt
- ❑ cp myfile.txt ../../

55

Exercise 2 – Organize the directories

Go back to your terminal and go to *practicals/files* directory

- **mkdir** - create directories
  - In *files* directory, create the directories *exercises*, and *play*.
  - In *files/play*, create directories *micro-array* and *grep*.
- - In *files/exercises*, create directories *ex1-explore*, *ex2-merge*, *ex3-retrieval*.
- **cp** - copy
  - make a copy of *files/uniprot-thioredoxin.fas* in directory *play/grep/*
- **mv** - move (from *files/* directory)
  - *rat_virus.dat* file to directory *exercises/ex1-explore/*
  - all 4 *seq1.fas* to *seq4.fas* files to directory *exercises/ex2-merge/*
  - *uniprot-thioredoxin.fas* to directory *exercises/ex3-retrieval/*
  - all 3 *arrayDat-1.csv* to *arrayDat-3.csv* files to directory *play/micro-array/*
- **mv** - rename
  - rename *exercises/ex3-retrieval/uniprot-thioredoxin.fas* to *thioredoxin.fas*

56

File organization

tree

```
.
├── files
│   ├── exam
│   ├── exercises
│   │   ├── ex1-explore
│   │   │   └── rat_virus.dat
│   │   ├── ex2-merge
│   │   │   ├── seq1.fas
│   │   │   ├── seq2.fas
│   │   │   ├── seq3.fas
│   │   │   └── seq4.fas
│   │   └── ex3-retrieval
│   │       └── thioredoxin.fas
│   ├── play
│   │   ├── grep
│   │   │   └── uniprot-thioredoxin.fas
│   │   └── micro-array
│   │       ├── arrayDat-1.csv
│   │       ├── arrayDat-2.csv
│   │       └── arrayDat-3.csv
│   ├── samExample.sam
│   └── uniprot-exam.fas
├── sandbox
```

```
├── sandbox
│   ├── 1stlevel
│   │   ├── 2ndlevel
│   │   │   ├── 3rdlevel
│   │   │   │   ├── 4thlevel
│   │   │   │   │   ├── pic1.png
│   │   │   │   │   ├── pic2.png
│   │   │   │   │   └── pic3.png
│   │   │   │   ├── butterfly.jpeg
│   │   │   │   ├── download.jpeg
│   │   │   │   ├── funny_hamster.jpeg
│   │   │   │   ├── red_rose.jpeg
│   │   │   │   └── tree.jpeg
│   │   │   ├── Q57921.xml
│   │   │   ├── Q57926.xml
│   │   │   ├── Q57941.xml
│   │   │   ├── Q57961.xml
│   │   │   ├── Q8TKQ6.xml
│   │   │   └── Q8TW28.xml
```

57

Find files or directories

find

- Very effective in finding files & directories
  find [path] [query type] 'query term'

Path

- '.' to search in the current directory and its sub-directories
- '/' (root directory) to search in all directories
- Use single or multiple '../' to search in parent directories
- '~' to search in your home directory

58

Find files or directories

Query types

- **-name** for a search based on the file name
  -iname same but case insensitive

```
find . -name '*.java'
```

- **-type f** or **-type d** to limit the search to files only or directories only
- **-size [+,-]n[scale]**, for a search based on the file size, where…
  + stands for 'larger than', **-** for 'smaller than', nothing for 'exact size',
  n indicates the numeric value,
  scale indicates the size: **c** for bytes; **k** for kilobytes; **M** for megabytes and **G** for gigabytes

```
find . -size +50M
```

- **-perm** [permission code]

```
find . -perm 644
```

59

Find files or directories

Tips

- Better to quote the query term with simple quotes
- Wildcards are possible
- ! placed after the path is the negative (*"that do no match the criteria"*)
- Options can be combined
- Boolean operators (AND, OR) can be used in combination with find

```
find . -name '*.mpeg' -and -size +30M (default)
find . -name '*.png' -or -name '*.jpg'
find . \( -name '*.png' -or -name '*.jpg' \)
-and -size +30M
```

When multiple arguments, possible to use regular expressions (-regex option)

60

File type determination

Reminder: file extensions **do not have a particular meaning** for the OS, thus might not correspond to the real file format

file

To check the **format** and detailed information of a file:

file filename(s)

- **text** - the file contains only printing characters and a few common control characters
- **executable** - the file contains the result of compiling a program
- **data** - anything else (data is usually binary or non-printable)

61

New file

touch

- creates a **new** empty file
  touch filename(s)

If *filename* already exists, this command does not erase file content, but updates its latest modification time

62

Quiz #5

Which command(s) list(s) all the files & directories from your home directory starting with 'exercise' in their name?

- find ~ -name 'exercise'
- file 'exercise'
- search -r 'exercise'
- find /home ’exercise* '
- find ~ -name ‘exercise* '

63

Quiz #6

Which command(s) give(s) information about the file type of a file named 'mystery'?

- man mystery
- find -type mystery
- which mystery
- file mystery

64

Exercise 3 – Find & identify

Go to directory *practicals/sandbox/* and answer the following questions:

- What is the format of file *create_million_line_file.sh*?
- Find files with either extension *.jpeg* or *.png*
- Find all *jpeg* files bigger than 10 ko

65

Archiving and extracting data

tar

- Creates, updates or expands archives

Options

- **-z** compresses using gzip program
- **-c** creates an archive
- **-x** extracts from an archive
- **-t** displays archive content (but does not extract it)

type of action

- **-v** displays information on the terminal window
- **-f** uses specified file
- **-m** set the current dates to extracted files

66

Archiving and extracting data

Highly recommended filename extensions:

- **.tar** for all archives
- **.gz** for gzip compressed archives

create an archive:	`tar -zcvf archive.tar.gz myfiles`
show archive content:	`tar -ztvf archive.tar.gz`
extract data from archive:	`tar -zxvf archive.tar.gz`
extract to another location:	`tar -zxvf archive.tar.gz -C location/…`
set the current date:	`tar -zxvmf archive.tar.gz`
upate an uncompressed archive	`tar -uf archive.tar new-files`

Tip: -f option must be placed at the last position because it requires argument (file name)

67

Quiz #7

Which command(s) uncompress(es) and untar(s) an archive called 'mydata.tar.gz'?

- ❑ tar -zxvf mydata.tar.gz
- ❑ untar mydata.tar.gz
- ❑ tar -zcvf mydata.tar.gz

68

In a nutshell

- Directory and file manipulation
  - mkdir, mv, cp, rm, rmdir
- Finding files and directories (find)
  - Wildcards characters for file search and display
- File determination (file)
- Archiving/extracting files (tar)

69

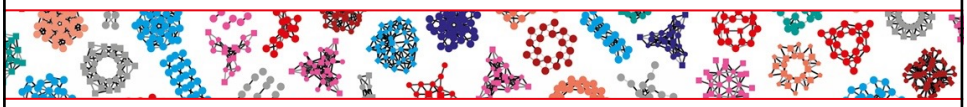
Outline

1. What is UNIX and why should biologists use it?
2. UNIX filesystem: navigation and usage
3. Processes & Jobs
4. Working with directories and files
5. Working with file content
6. Optional exam

Exercises are integrated in chapters

70

Working with file content



Swiss Institute of Bioinformatics

71

Line by line

Rule:

Content search is performed

line by line

72

File content statistics

wc

wc option filename(s)

Options

- **-l** prints the number of lines
- **-w** prints the number of words
- **-c** prints the number of bytes
- **-m** prints the number of characters

Without options, **wc** outputs the number of lines, words and bytes

73

File content display: head & tail

head

- aims at quickly showing the **beginning of a file**

head *myfile.txt* -> displays the first 10 lines (default)

head *-2 myfile.txt* -> displays the first 2 lines

head *-n-2 myfile.txt* -> displays all the lines, except the last 2

tail

- is similar to **head**, but shows the **last lines** of a file

tail *myfile.txt* -> displays the last 10 lines

tail *-2 myfile.txt* -> displays the last 2 lines

tail *-n+2 myfile.txt* -> displays all lines except the first (useful e.g. to remove header line)

74

File content display: cat & less

cat displays the content of a text file

When several files are passed as arguments, **cat** concatenates and displays their content, thus can be used to merge the content of several files

75

File content display: cat & less

less

- displays the content of a text file **one screen at a time**

Option

-N shows line numbers at the left of each line

Useful commands to be used inside the text file:

- **[space-bar]**, shows the next page, one screen at a time
- **up/down arrow keys**, moves the content backward/forward one line at a time
- **<** and **>** to reach the beginning and the end of the document
- **/mouse**: search the string "mouse"
- **q** to quit

76

Quiz #8

Which command(s) show(s) the first 10 lines of a file on the screen?

- ❑ tail
- ❑ head -10
- ❑ print -10
- ❑ head

77

Quiz #9

The file 'uniprot-mouse-virus.fasta' contains more than 7000 protein sequences. What will output the following command ?

```
head uniprot-mouse-virus.fasta | wc -l
```

- ❑ 6990
- ❑ 0
- ❑ 7000
- ❑ 10

78

Exercise 4 – Content display

Locate the file *uniprot-thioredoxin.fas*, navigate to it, then use the following commands:

- **head, tail**
  - display the first 10 lines
  - display the last 5 lines
- **wc**
  - count only the number of lines
  - count only the number of words
- **cat**
  - view the file with *cat*. Why is this not the most adapted program here? Indicate another usage of *cat*?
- **less**
  - view the file with *less* (add the option -*N* to view lines number), navigate with space bar / arrows
  - search for pattern *isoform* using '/ ', then navigate through matching patterns with keys n and N
  - quit with 'q'

79

Text editors

Interactive use –> no need for text edition

Programmning -> need for text edition

Several programs available: Sublime Text, Visual Studio Code, etc

Vim is a text editor integrated into most terminals

Nano

80

Standard streams

UNIX has three standard streams

- **Standard input** - **stdin**, where programs receive data from:
  - from the **keyboard**
  - from a **file** (using <)
- **Standard output** - **stdout**, where programs write their output data to:
  - **printed** on the terminal window
  - **written** in a file (using > or >>)
- **Standard error** - **stderr**, to output error messages
  - **printed** on the terminal window

81

Standard streams

Example of **standard input** from file

```
command < ID_B.txt
```

< not mandatory with most programs

82

Standard streams

Example of standard **output to a file**

Command **>** *filename*

- if the file does not exist, it is created
- If it does, its content is **overwritten**

Command **>>** *filename*

- if the file does not exist, it is created (same function as **>**).
- If it does, the output is **added at the end** of the file content.

83

Standard streams

Output redirection: the **|** (pipe):

```
userA$ tr " " "\n" < go-data.csv | grep ^GO: | sort | uniq -c | sort -nr | head -4
```

useful to pass directly the output from one program as the input of another **without creating intermediate files**

84

Delimiter-Separated Values (DSV)

Free-text (not DSV)

```
Hello, I will be very busy the next couple of months: I started a
course in Bioinformatics!
```

Comma/Semicolon-separated values (CSV)

```
Entry,Entry_name,Protein_names
Q8QMT5,A18_CWPXB,Transcript termination protein A18
Q80DV6,A18_CWPXG,Transcript termination protein A18
```

Tab-separated values (TSV)

```
Entry          Entry name     Protein name
Q8QMT5         A18_CWPXB      Transcript termination protein A18
Q80DV6         A18_CWPXG      Transcript termination protein A18
```

text identification, retrieval and manipulation is more efficient when structured with delimiters

85

Delimiter-Separated Values (DSV)

Biological sequences are often represented in a so-called **FASTA** format:

- a first line starting with '>' and free text / comments (header line)
- main lines composed of one-letter code sequences

```
>sp|Q1PE49|4ON1_ARATH Protein At-4/1 OS=Arabidopsis thaliana
MAATSDEQMNLLLSSFDQIYEDFKIGLNEINVYRSKSNVESSRREVLEISNKNLKEENERL
KKLYTESLNNFADQLEHRTKCHSLKEELKRVNDENKSKEHEHRNALESLRQKHVTKVEE
```

86

Newline character

structured formats → a 'newline' character* at the end of each line, which **IS NOT** the same for all OS

- **\n** -> Unix, Mac OS X
- **\r** -> Mac OS (before X)
- **\r\n** -> Windows OS

*use command *od -c* to view them

conversion is sometimes necessary

87

Converting file content

To convert newline characters…

tr (for 'translate') can do this in a single command line:

tr *'\r' '\n' < myfile > mynewfile*

tr is very convenient to change a delimiter, e.g. from TSV to CSV

If you want to replace words or phrases, the command **sed** is required (not discussed during this course)

88

Sorting file content

sort

- used to sort lines of text files
- default is alphabetical order

Options

- **-n** for a numeric sort
- **-r** for reverse sort
- **-u** for unique lines (keeps one instance when identical lines)
- **-k** sorts via a key (a column number for example)
- **-t** to define a key separator (e.g. a |). Default is space, but tab is recognized too

sort -k is useful if you want to sort according to only some parts of a line, typically columns

89

Searching / filtering file content

uniq

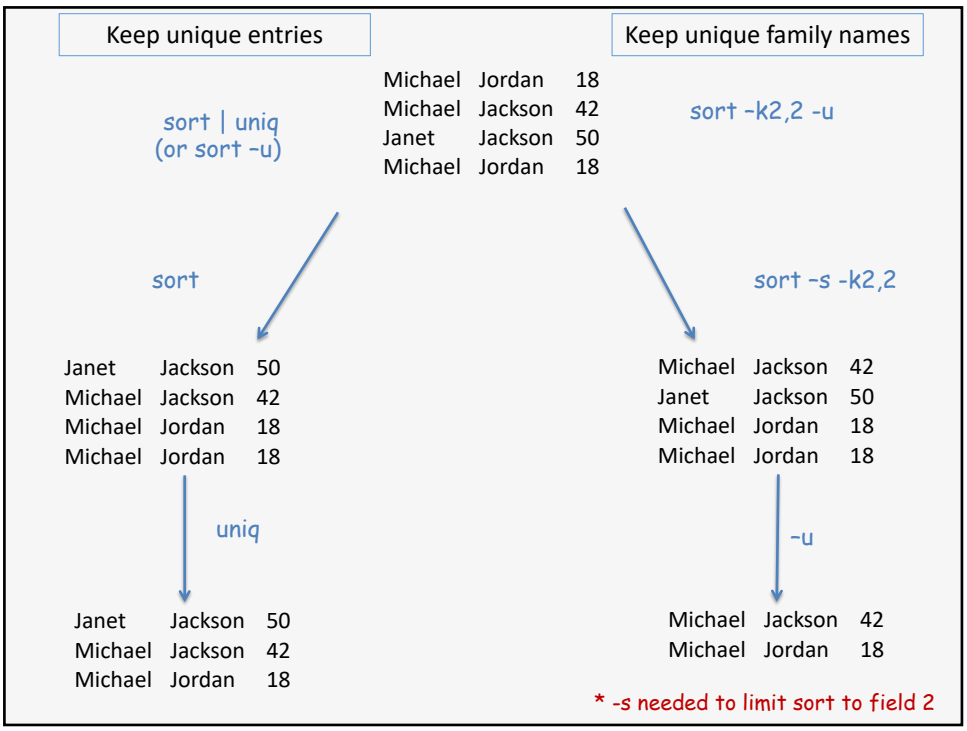
- reports **unique lines** by suppressing multiple occurrences of **identical** lines if **sequential**. Therefore, it is typically used after **sort**

Options

- **-d**, outputs only lines that are repeated (duplicated or more) in the input
- **-c**, precedes each reported line with the **number of occurrences** found
- **-i**, ignores the differences in case when comparing the lines
- **-u**, only prints unique lines

sort | uniq and sort -u have the same function: suppression of multiple occurrence of identical lines and ordering, but sort has an option (-k) to sort on specific key (e.g. a column) instead of the entire line.

90



91

Filtering / extracting file content

cut is used to extract selected parts of lines (like columns). Convenient to manipulate delimiter-separated value files

Options

- **-c**, selection based on characters positions. This is used when cut has to be performed at a fixed position (e.g. to remove a fixed header)
- **-d**, type of delimiter (default is 'tab'), e.g. ';' ',' ' ' or any character
- **-f**, selection based on fields in DSV files.
  single field number (e.g. *-f 1*) , multiple field numbers (e.g. *-f 1,4*) or a range (e.g. *-f 1-4*) must be indicated. Fields are determined by delimiters if not TAB

NB: cut -f 1 | sort -u and sort -k1,1 -u are almost identical but the former will display only the extracted column

92

Comparing file content

diff compares **2** files and outputs the differences.

Useful to compare:

- 2 files with the same name in different directories
- different versions of the same file

Options

- **-b**, ignores differences in amount of white space
- **-B**, ignore differences of blank lines
- **-s**, reports when two files are identical (it does not by default)
- **-q**, reports only whether two files differ, no details of the differences

93

Merging file content

cat

- display function -> can be used to concatenate content from several files
- concatenation does not include any blank or newline

Options

- **-s** , squeezes multiple adjacent empty output lines
- **-b**, numbers nonempty output lines

paste

- can merge columns from different files

Options

-d allows to introduce delimiters between columns (':' ',' '-')

94

Exercise 5 – Merge data in columns

Context: three files containing replicates from a microarray data experiment

Input files: */play/micro-array/arrayDat-1.csv* to *arrayDat-3.csv*

Objective: merge values from 3 files (keep one *ProbeID* column) into one single file, organised in tab-delimited values

95

Input files

```
ProbeID;Sample1
1007_s_at;10.93
1053_at;8.28
117_at;3.31
121_at;4.42
1255_g_at;1.8
1294_at;2.69
1316_at;3.26
1320_at;5.54
1405_i_at;4.29
1431_at;2.19
1438_at;4.05
1487_at;7.92
1494_f_at;2.64
1552256_a_at;8.72
1552257_a_at;6.81
1552258_at;2.63
1552261_at;2.69
1552263_at;2.94
1552264_a_at;7.02
1552266_at;2.94
1552269_at;3.87
1552271_at;2.93
1552272_a_at;3.1
1552274_at;2.67
1552275_s_at;2.26
1552276_a_at;2.95
1552277_a_at;9.16
1552278_a_at;3.14
1552279_a_at;4.08
1552280_at;2.05
1552281_at;3.75
1552283_s_at;3.32
1552286_at;5.25
arrayDat-1.csv
```

```
ProbeID;Sample2
1552390_a_at;2.76
1552389_at;3.4
1552388_at;2.61
1552386_at;1.35
1552384_a_at;2.2
1552383_at;4.47
1552381_at;2.44
1552379_at;1.76
1552378_s_at;4.38
1552377_s_at;2.46
1552375_at;2.29
1552373_s_at;1.99
1552372_at;1.66
1552370_at;5.31
1552368_at;2.47
1552367_a_at;4.14
1552365_at;7.19
1552364_s_at;6.97
1552362_a_at;3.35
1552360_a_at;3.71
1552359_at;2.19
1552355_s_at;1.99
1552354_at;3.06
1552349_a_at;3.08
1552348_at;2.48
1552347_at;4.82
1552344_s_at;5.97
1552343_s_at;3.13
1552340_at;2.02
1552338_at;3.12
1552337_s_at;2.7
1552335_at;2.64
1552334_at;2.43
arrayDat-2.csv
```

```
ProbeID;Sample3
1007_s_at;11.19
1053_at;8.06
117_at;3.13
121_at;4.46
1255_g_at;1.75
1294_at;2.37
1316_at;3.12
1320_at;4.7
1405_i_at;5.05
1431_at;2.12
1438_at;4.42
1487_at;8.24
1494_f_at;2.53
1552256_a_at;9.47
1552257_a_at;7.03
1552258_at;2.08
1552261_at;2.64
1552263_at;3.22
1552264_a_at;6.93
1552266_at;4.06
1552269_at;3.34
1552271_at;2.44
1552272_a_at;2.34
1552274_at;3.26
1552275_s_at;2.58
1552276_a_at;2.44
1552277_a_at;8.77
1552278_a_at;4.01
1552279_a_at;4.51
1552280_at;2.42
1552281_at;3.38
1552283_s_at;3.37
1552286_at;5.13
arrayDat-3.csv
```

96

Expected output file

```
ProbeID Sample1 Sample2 Sample3
117_at  3.31    3.41    3.13
121_at  4.42    4.32    4.46
1007_s_at       10.93   11.44   11.19
1053_at 8.28    7.54    8.06
1255_g_at       1.8     1.7     1.75
1294_at 2.69    2.72    2.37
1316_at 3.26    2.8     3.12
1320_at 5.54    6.19    4.7
1405_i_at       4.29    5.3     5.05
1431_at 2.19    1.99    2.12
1438_at 4.05    4.59    4.42
1487_at 7.92    7.98    8.24
1494_f_at       2.64    2.48    2.53
1552256_a_at    8.72    8.73    9.47
1552257_a_at    6.81    6.56    7.03
1552258_at      2.63    2.49    2.08
1552261_at      2.69    2.83    2.64
1552263_at      2.94    3.48    3.22
1552264_a_at    7.02    7.1     6.93
1552266_at      2.94    3.92    4.06
1552269_at      3.87    4.57    3.34
1552271_at      2.93    3.04    2.44
1552272_a_at    3.1     2.88    2.34
1552274_at      2.67    3.18    3.26
1552275_s_at    2.26    2.78    2.58
1552276_a_at    2.95    3       2.44
1552277_a_at    9.16    8.83    8.77
1552278_a_at    3.14    3.16    4.01
1552279_a_at    4.08    4.33    4.51
1552280_at      2.05    1.86    2.42
1552281_at      3.75    3.37    3.38
1552283_s_at    3.32    3.45    3.37
1552286_at      5.25    4.88    5.13
final.tsv
```

97

Step 0

- have a look inside the files using *less* or *head*, to check out how they are made

Step 1: convert comma-separated values to tab-separated values

- **tr**
- convert the delimiters from *;* to *\t* (tab)
- and save the new format into a new file with a .tsv extension (e.g. *arrayDat-1.tsv*)

98

Step 2

Input files: *arrayDat-1.**tsv*** to *arrayDat-3.**tsv***

- **sort**
- before merging, the *ProbeID* column of each file needs to be sorted. Beware, header line must remain at the top
- for each file, order lines according to column 1
- save the output to temporary files *tmp1.tsv* to *tmp3.tsv*

99

Step 3

Input files: *tmp1.tsv* to *tmp3.tsv*

- **cut**
- for each file, extract column 1
- save the output to temporary file *tmp1a.tsv* to *tmp3a.tsv*

100

Step 4

Input files: *tmp1a.tsv* to *tmp3a.tsv*

- **diff**
- compare *tmp1a.tsv* and *tmp2a.tsv*
- compare *tmp2a.tsv* and *tmp3a.tsv*

101

Step 5

Input files: *tmp1.tsv* to *tmp3.tsv*

- **paste**
- paste columns from all files to have probeID, sample1, sample2, sample3 (better to do in 2 steps -> save intermediate results)

102

Output: all samples are grouped in a same tab-separated file (tsv) with the header line on top

ProbeID	Sample1	Sample2	Sample3
117_at	3.31	3.41	3.13
121_at	4.42	4.32	4.46
1007_s_at	10.93	11.44	11.19
1053_at	8.28	7.54	8.06
1255_g_at	1.8	1.7	1.75
1294_at	2.69	2.72	2.37
1316_at	3.26	2.8	3.12
1320_at	5.54	6.19	4.7
1405_i_at	4.29	5.3	5.05

103

Searching file content

grep

- for **G**lobal **R**egular **E**xpression **P**rint - searches file(s) for words or patterns and return matching lines
  grep [options] 'pattern' file(s)

Options

- **-c**, displays the **number** of resulting lines instead of the lines themselves
- **-i**, performs a **case insensitive** search
- **-v**, displays lines that **DO NOT** match the pattern
- **-o**, returns only the pattern, not the entire line
- **-n**, adds the **line number** in front of the result lines
- **-r**, search all files **recursively** inside each directory

104

Regular Expressions

Characters used to search a pattern, example:

^a	‘a’ at the beginning of a line
a$	‘a’ at the end of a line
^$	to express a blank line
^.$	any character on a line
[abc]	either character
[a-z]	a lowercase character
[0-9]	any figure
[0-59]	0, 1, 2, 3, 4, 5 or 9
[^0-9]	any character except a figure

http://www.grymoire.com/Unix/Regular.html

105

Quiz #10

Which command(s) count(s) the number of lines matching the string 'task' in the file 'exercise.txt'?

- grep -v 'task' exercise.txt
- grep -iv 'task' exercise.txt
- grep -c 'task' exercise.txt
- grep -n 'task' exercise.txt

106

Exercise 6a – Information retrieval

Input file: *play/grep/uniprot-thioredoxin.fas*

Have a look at file content

- **grep**
- How many sequences? (count header lines)
- How many entries from 'Staphylococcus'?
- How to display header lines not from 'Staphylococcus'?

107

Exercise 6b – Information retrieval

Input: One fasta file (***thioredoxin.fas***) containing > 3000 sequences

Objective: Display the 10 most frequent genus represented and their frequency

Source: */exercises/ex3-retrieval/*

Tip: Take advantage of controlled vocabulary, like organism abbreviation (OS=)

Step 1: Isolate the header lines

Step 2: Retrieve genus name

108

Step 3: Sort and keep unique genus

Step 4: Order by occurrence and show the 10 most frequent

```
168 Arabidopsis
166 Escherichia
163 Bacillus
152 Homo
141 Staphylococcus
134 Mus
111 Oryza
 84 Salmonella
 83 Rattus
 72 Mycobacterium
```

109

In a nutshell

- UNIX allows to manipulate file content, such as order lines, cut fields, merge file content
- UNIX allows complex searches to retrieve filtered data
- Complex commands might be written into scripts
- Note that other languages are better suited to this sort of complex manipulations, like AWK, perl, python, or R

110

Next…

- The exam doc is now available on the course page (again you need to be identified with the common account)
- Please send your results to Robin and myself before tomorrow evening…
- The Feedback form is available (again you need to be identified with the common account). Thank you it will help us, in particular due to this special online streamed format.

111

Next step

UNIX scripting for Life Scientists

By Thomas Junier & Robin Engler

7-8 Novmeber 2022

Bern

https://www.sib.swiss/training/course/20221107_ADVU

112

Further resources

Our e-learning tutorial:

https://edu.sib.swiss/course/view.php?id=82

Resources that could be of interest:

http://en.wikipedia.org/wiki/List_of_Unix_commands
http://www.catonmat.net/projects/cheat-sheets/
https://www.bits.vib.be/index.php/training/124-linux-for-bioinformatics
https://ubuntudanmark.dk/filer/fwunixref.pdf

113

Acknowledgements

- Diana Marek
- Vassilios Ioannidis
- Volker Flegel
- Frédéric Schütz
- Heinz Stockinger
- Ivan Topolsky
- Alex Smith

Thank you for your attention!

http://sib.swiss/training
training@sib.swiss

Swiss Institute of Bioinformatics

114